第3章 栈和队列

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

3.1 栈和队列的定义和特点

3.2 栈的表示和操作的实现

3.3 栈的案例分析

3.4 栈和递归

3.5 队列的表示和操作的实现

3.4： 栈和递归

递归的定义

- 若一个对象部分地包含它自己，或用它自己给自己定义，则称这个对象是递归的
- 若一个过程直接或间接地调用自己，则称这个过程是递归的过程
  - 例如：递归求n的阶乘

```
long Fact ( long n ) {
    if ( n == 0) return 1;
    else return n * Fact (n-1);
}
```

3.4： 栈和递归

- 以下三种情况常常用到递归方法
  - 递归定义的数学函数
  - 具有递归特性的数据结构
  - 可递归求解的问题

3.4： 栈和递归

1 递归定义的数学函数

- 阶乘函数

$$Fact(n)=\begin{cases}1 & \text{若n}=0\\ n\cdot Fact(n-1) & \text{若n}>0\end{cases}$$

- 2阶Fibonacci数列：

$$Fib(n)=\begin{cases}1 & \text{若n}=1\text{或}2\\ Fib(n-1)+Fib(n-2) & \text{其它}\end{cases}$$

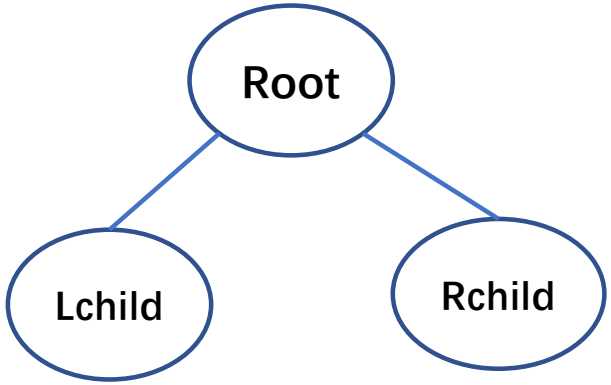
3.4： 栈和递归

- 以下三种情况常常用到递归方法
  - 递归定义的数学函数
  - 具有递归特性的数据结构
  - 可递归求解的问题

3.4： 栈和递归

2 具有递归特性的数据结构

- 二叉树



- 广义表

A = （a, A）

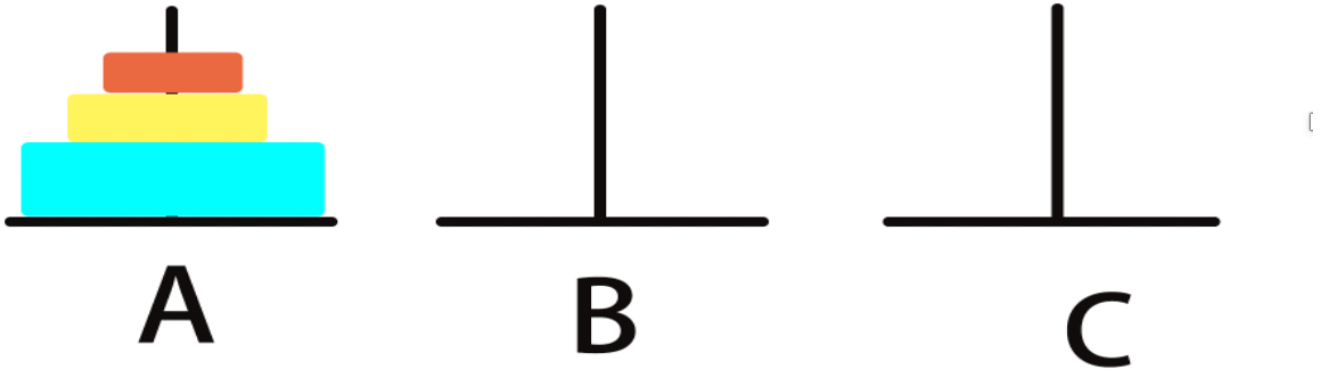
3.4： 栈和递归

- 以下三种情况常常用到递归方法
  - 递归定义的数学函数
  - 具有递归特性的数据结构
  - 可递归求解的问题

3.4： 栈和递归

3 可递归求解的数学问题

- 汉诺塔问题



3.4： 栈和递归

- 以下三种情况常常用到递归方法
  - 递归定义的数学函数
  - 具有递归特性的数据结构
  - 可递归求解的问题

3.4： 栈和递归

递归问题——用分治法求解

分治法：对于一个较为复杂的问题，能够分解成几个相对简单的且解法相同或相似的子问题来求解

必备的三个条件

1、能够将一个问题转变成一个新问题，而新问题与原问题的解法相同或类同，不同的仅是处理的对象，且这些处理对象的变化有规律的

2、可以通过上述转化而使问题简化

3、必须有一个明确的递归出口，或称递归的边界

3.4： 栈和递归

分治法求递归问题算法的一般形式：

```
void   p (参数表) {
       if    (递归结束条件) 可直接求解步骤；-----基本项
       else  p (较小的参数) ；------归纳项
}
```

例如：

```
long Fact ( long n ) {
    if ( n == 0) return 1;          //基本项
    else return n * Fact (n-1);     //归纳项
}
```

3.4： 栈和递归

- 函数调用过程

调用前、系统完成

（1）将实参、返回地址等传递给被调用函数
（2）为被调用函数的局部变量分配存储区
（3）将控制转移到被调用函数的入口

调用后，系统完成

（1）保存被调用函数的计算结果
（2）释放被调用函数的数据区
（3）依照被调用函数保存的返回地址将控制转移到调用函数

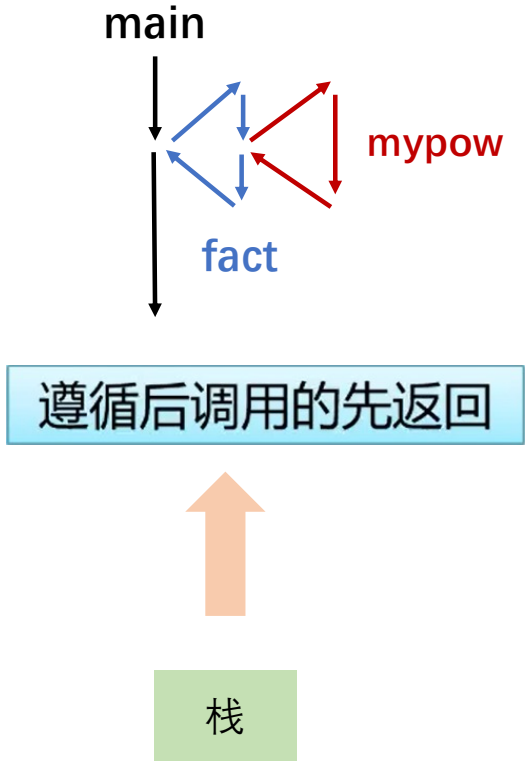
3.4： 栈和递归

当多个函数嵌套调用时：

```
int main(void) {
    ......
    y = fact(3);
    ......
}

double fact(int n) {
    ......
    z = mypow(3.5, 2);
    ......
}

double mypow(double x, in n) {
    ......
}
```

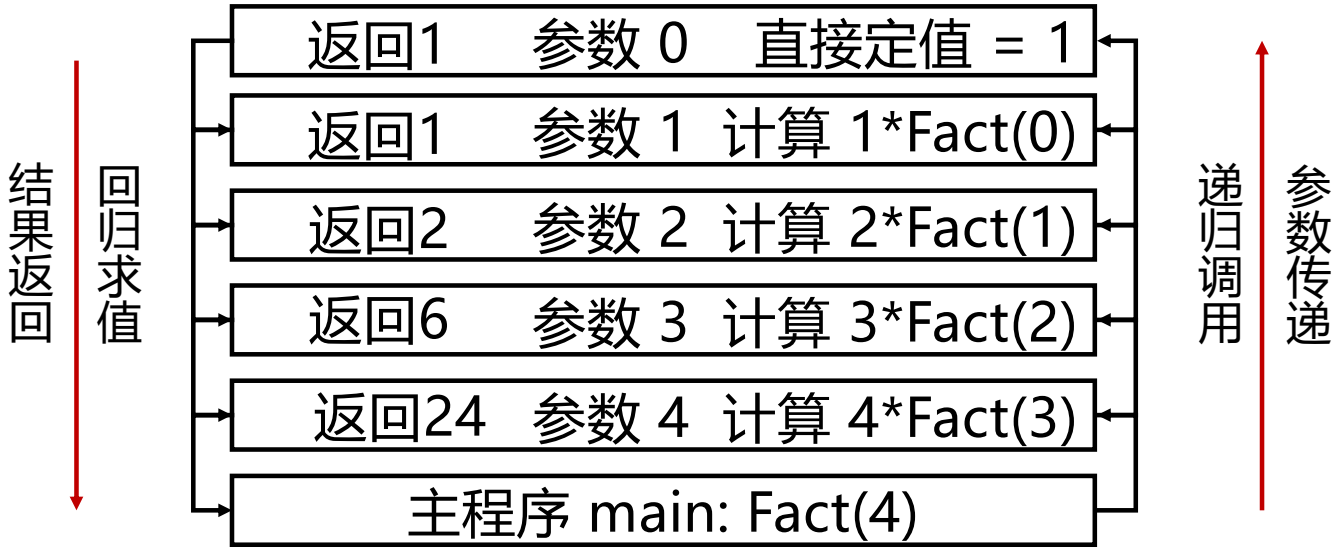


3.4： 栈和递归

求解阶乘n!的过程

主程序 main: Fact(4)

```
if ( n == 0 ) return 1;
else return n * Fact (n-1);
```



3.4： 栈和递归

可递归函数调用的实现

“层次” 主函数 0层

第一次调用 1层

……

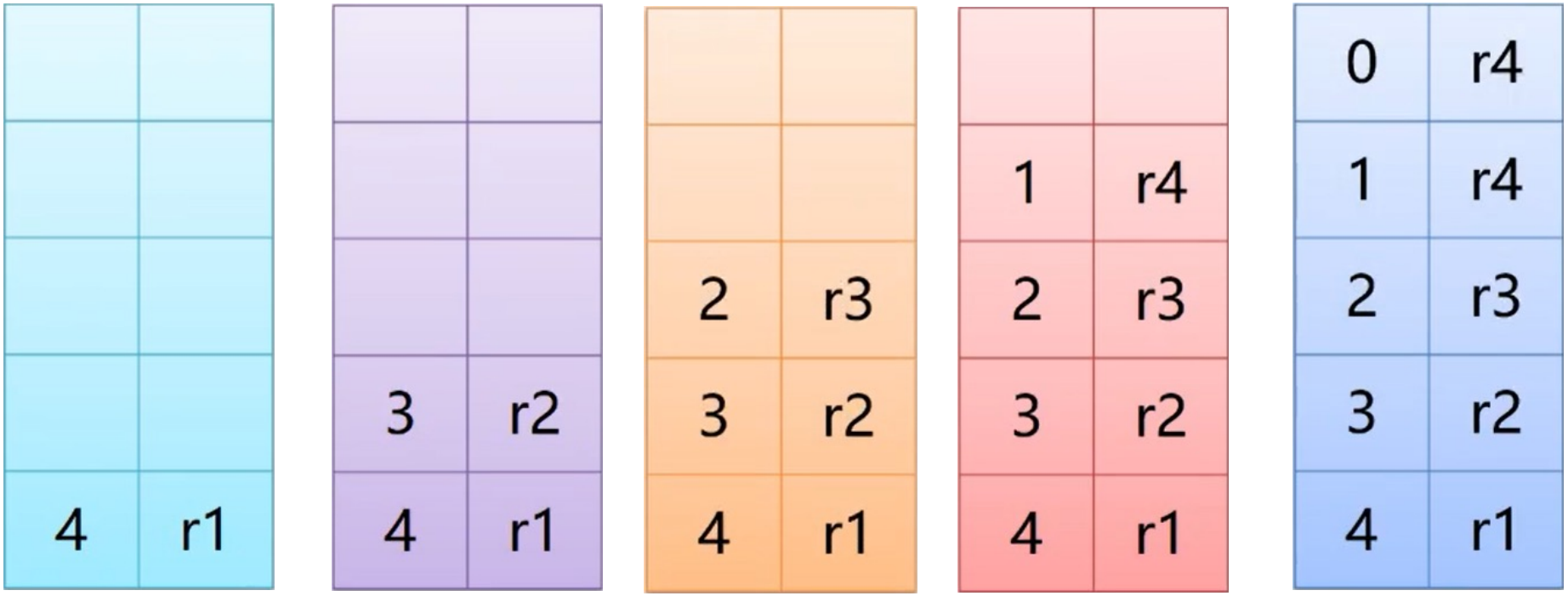
第i次调用 i层

“递归工作栈”——递归程序运行期间使用的数据存储区

“工作记录” → 实参，局部变量，返回地址

3.4： 栈和递归

- 进行fact(4)调用的系统栈的变化状态



3.4： 栈和递归

递归的优缺点

优点：结构清晰，程序易读

缺点：每次调用要生成工作记录，保存状态记录，入栈；返回时要出栈，回复状态信息。时间开销大

第3章 栈和队列

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

3.1 栈和队列的定义和特点

3.2 栈的表示和操作的实现

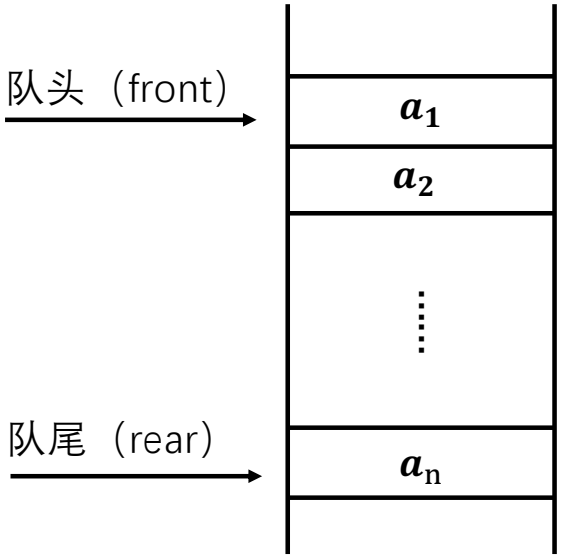
3.3 栈的案例分析

3.4 栈和递归

3.5 队列的表示和操作的实现

3.5： 队列的表示和操作的实现

队列示意图



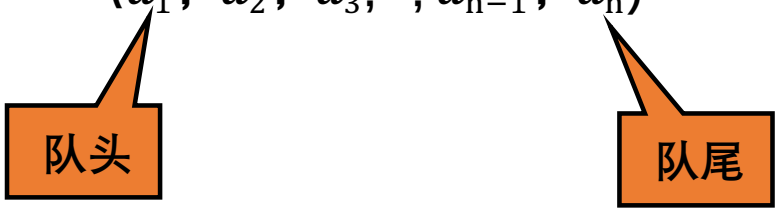
3.5： 队列的表示和操作的实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

相关术语

- 队列（queue）是仅在表尾进行插入操作，在表头进行删除操作的线性表
- 表尾即a_n端，称为队尾；表头即a_1端，称为队头
- 它是一种先进先出（FIFO）的线性表

例如：队列 Q = $(a_1, a_2, a_3, \cdots, a_{n-1}, a_n)$



插入元素称为入队；删除元素称为出队。

队列的存储结构为链队或顺序队（常用循环顺序队）

队列的定义和特点

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

队列的相关概念

1.定义	限定只能在表的一端进行插入运算，在表的另一端进行删除运算的线性表（头删尾插）
2.逻辑结构	与线性表相同，仍为一对一关系
3.存储结构	用顺序队或链队均可，但以循环顺序队更常见
4.运算规则	只能在队首和队尾运算，且访问节点时依照先进先出（FIFO）的原则
5.实现方式	关键是掌握入队和出队操作，具体实现依顺序队或链队的不同而不同

队列的常见应用

- 由于队列的操作具有先进先出的特性，使得队列成为程序设计中解决类似排队问题的有用工具。

- 脱机打印输出：按申请的先后顺序依次输出
- 多用户系统中，多个用户排成队，分时地循环使用CPU和主存
- 按用户的优先级排成多个队，每个优先级一个队列
- 实时控制系统中，信号按接受的先后顺序依次处理
- 网络电文传输，按到达的时间先后循序依次进行

3.5.1： 队列的抽象数据类型定义

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

ADT Queue{

数据对象： $D=\{a_i | a_i \in ElemSet, i=1,2,...,n, n\geq 0\}$

数据关系： $R=\{<a_{i-1},a_i> | a_{i-1},a_i \in D, i=2,...,n\}$ 约定其中a_1端为队列头， a_n 端为队列尾。

基本操作：

InitQueue(&Q)	操作结果：构造空队列Q
DestroyQueue(&Q)	条件：队列Q已存在； 操作结果：队列Q被销毁
ClearQueue(&Q)	条件：队列Q已存在； 操作结果：将Q清空
QueueLength(Q)	条件：队列Q已存在 操作结果：返回Q的元素个数，即队长
GetHead(Q, &e)	条件：Q为非空队列 操作结果：用e返回Q的队头元素
EnQueue(&Q, e)	条件：队列Q已存在 操作结果：插入元素e为Q的队尾元素
DeQueue(&Q, &e)	条件：Q为非空队列 操作结果：删除Q的队头元素，用e返回值

..... 还有将队列置空、遍历队列等操作.....

} ADT Queue

3.5.2： 队列的顺序表示和实现

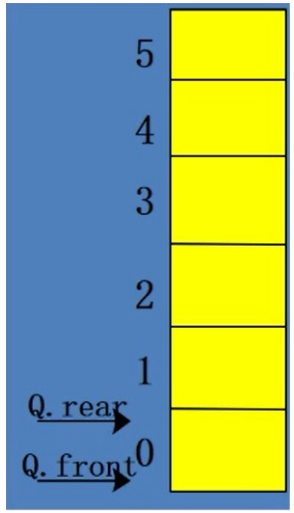
杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

- 队列的物理存储可以用顺序存储结构，也可用链式存储结构。相应地，队列的存储方式也分为两种，即顺序队列和链式队列。

- 队列的顺序表示——用一维数组base [MAXQSIZE]

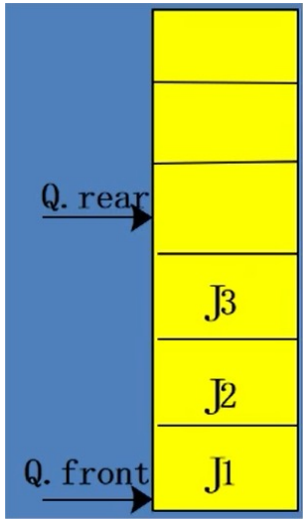
```
#define MAXQSIZE  100   //最大队列长度
Typedef struct {
   QElemType *base;  //初始化的动态分配存储空间
   int  front;              //头指针
   int  rear;               //尾指针
}SqQueue;
```

3.5.2： 队列的顺序表示和实现



初始：

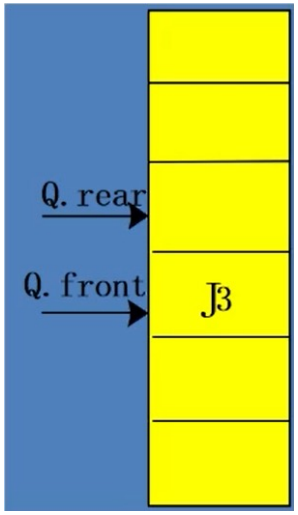
front=rear=0



J1、J2、J3入队

入队：base[rear]=x;

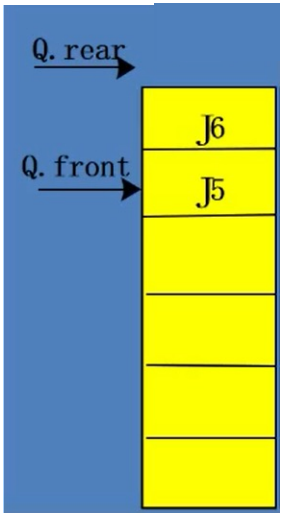
rear++;



J1、J2出队

出队：x=base[front]; front++;

空队标志：front= =rear

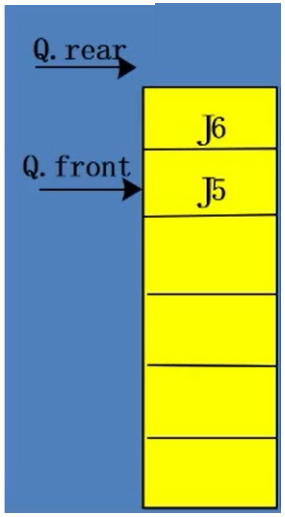


J4、J5、J6入队

J3、J4出队

还可以入队吗?

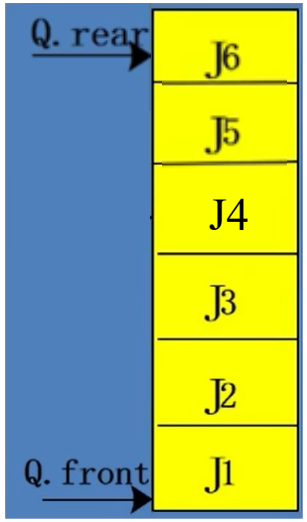
3.5.2： 队列的顺序表示和实现



思考：存在什么问题？？？

设数组大小为MAXQSIZE

rear=MAXQSIZE时，发生溢出



若front=0

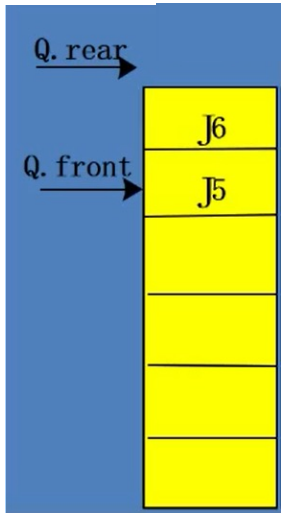
rear=MAXQSIZE时

再入队—真溢出

front≠0

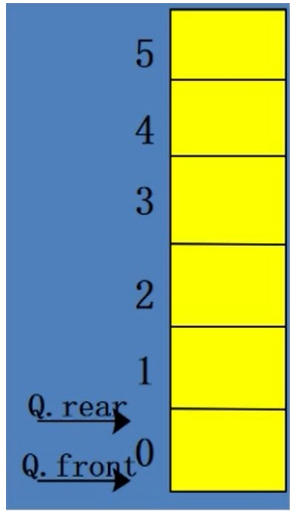
rear=MAXQSIZE时

再入队—假溢出



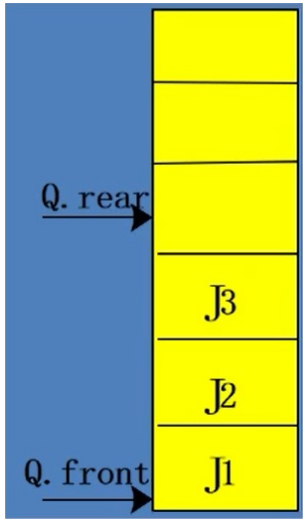
3.5.2： 队列的顺序表示和实现

3.5.2： 队列的顺序表示和实现



初始：

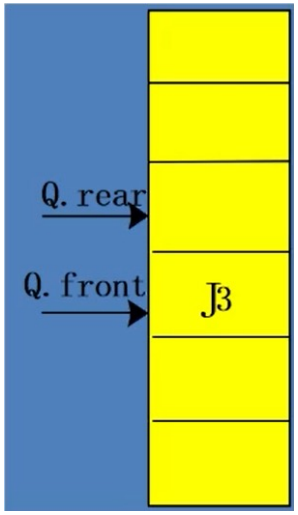
front=rear=0



J1、J2、J3入队

入队：base[rear]=x;

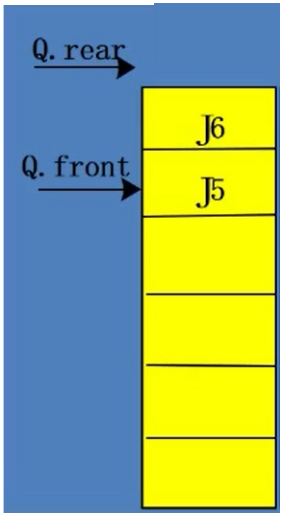
rear++;



J1、J2出队

出队：x=base[front]; front++;

空队标志：front= =rear



J4、J5、J6入队

J3、J4出队

还可以入队吗?

3.5.2： 队列的顺序表示和实现

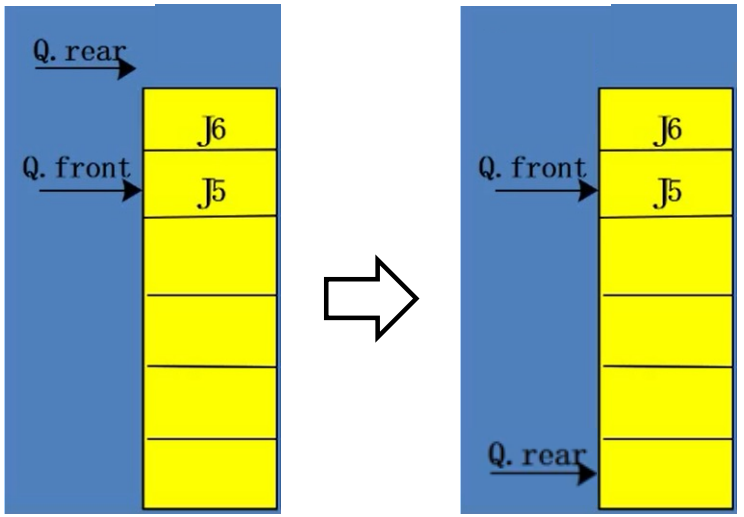
解决假上溢的方法

1、将队中元素依次向队头方向移动。

缺点：浪费时间。每移动一次，队中元素都要移动

2、将队空间设想成一个循环的表，即分配给队列的M个存储单元可以循环使用，当rear为maxsize时，若向量的开始端空着，又可以从头使用空着的空间。当front为Maxsize时也是一样。

MAXQSIZE = 6



3.5.2： 队列的顺序表示和实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

解决假上溢的方法——引入循环队列

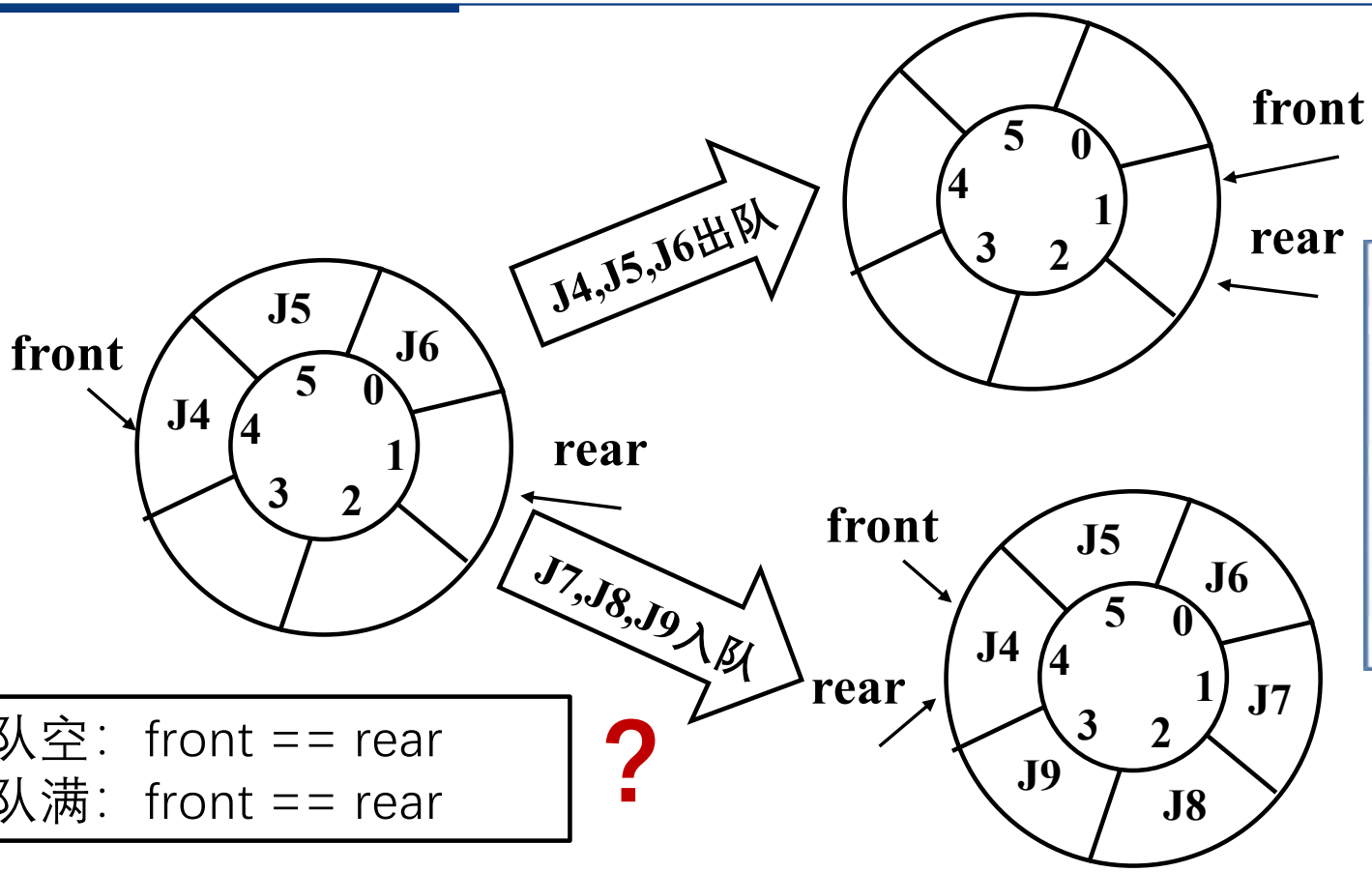
Base[0]接在base[MAXQSIZE – 1]之后，若rear+1=M，则令rear=0;

实现方案：利用模（mod ,C 语言中：%）运算。

插入元素：Q.base[Q.rear] = x;
Q.rear = (Q.rear+1)%MAXQSIZE;

删除元素：x = Q.base[Q.front];
Q.front = (Q.front+1)%MAXQSIZE;

3.5.2： 队列的顺序表示和实现

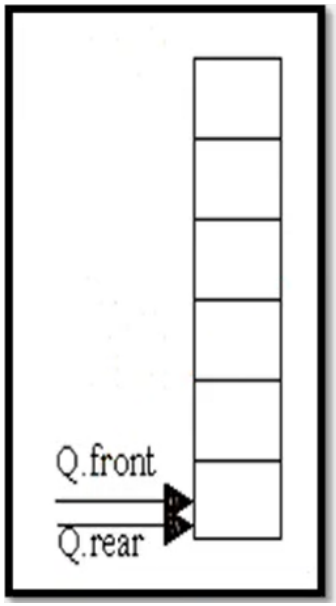


队空： front == rear
队满： front == rear

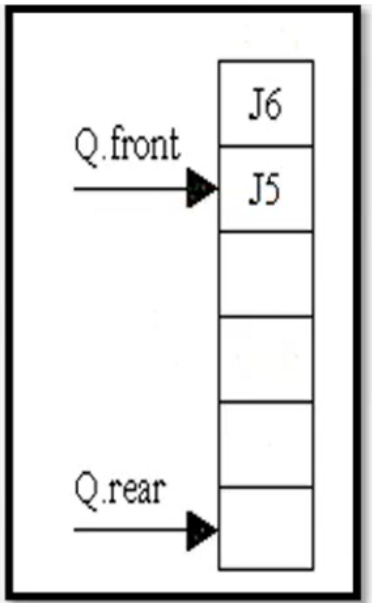
?

解决方案：
1.另外设一个标志以区别队空、队满
2.另设一个变量，记录元素个数
3.少用一个元素空间

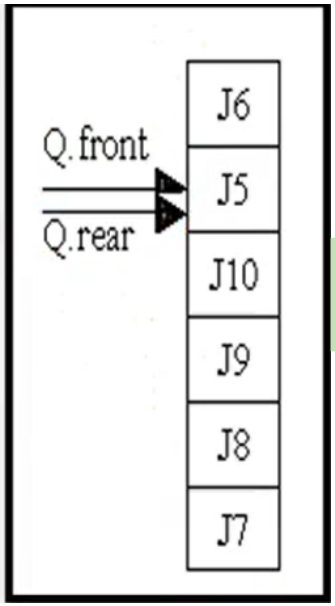
3.5.2： 队列的顺序表示和实现



空队

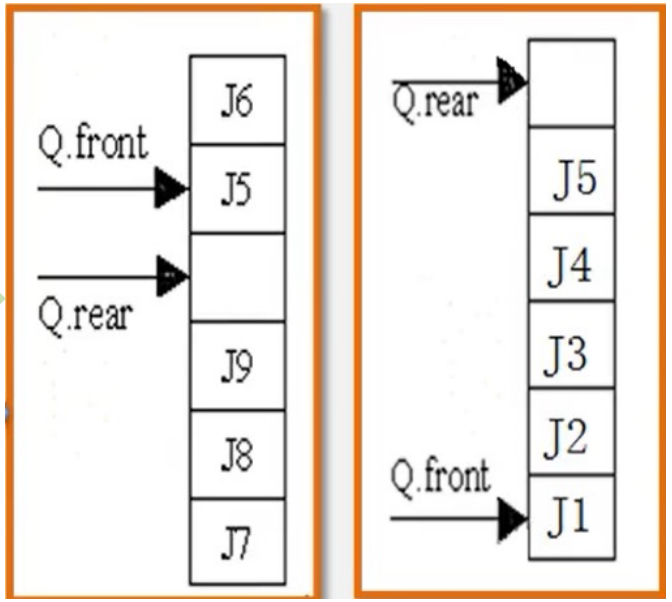


一般情况



队满：队尾指针和队头指针重合

少用一个空间元素



队满：队尾指针+1就和队头指针重合

队空： front == rear

队满： (rear+1)%MAXQSIZE==front

3.5.2： 队列的顺序表示和实现

循环队列的类型定义

```
#define MAXQSIZE 100 //最大队列长度
typedef struct {
    QElemType *base; // 动态分配存储空间
    int front;       // 头指针，若队列不空，指向队列头元素
    int rear;        // 尾指针，若队列不空，指向队列尾元素的下一个位置
 } SqQueue;
```

3.5.2： 队列的顺序表示和实现

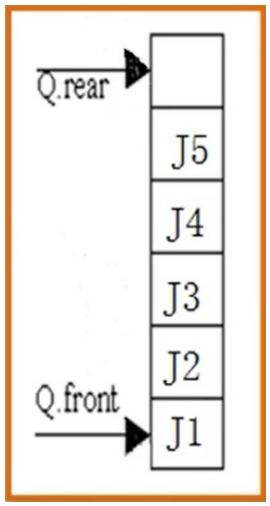
循环队列的操作——队列的初始化

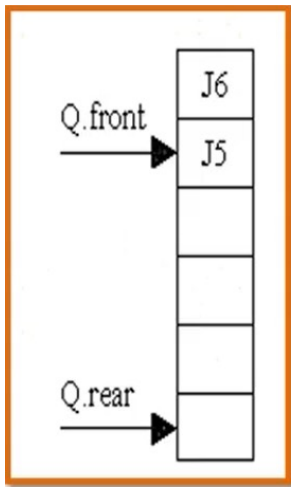
```
Status InitQueue (SqQueue &Q){
    Q.base =new QElemType[MAXQSIZE]      //分配数组空间
    //Q.base = (QElemType*)
    malloc(MAXQSIZE*sizeof(QElemType));
    if(!Q.base) exit(OVERFLOW);              //存储分配失败
    Q.front=Q.rear=0;                  //头指针尾指针置为0，队列为空
    return OK;
}
```

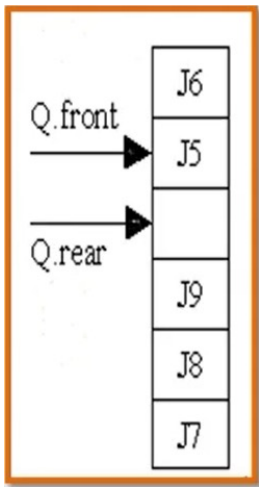
3.5.2： 队列的顺序表示和实现

循环队列的操作——求队列的长度

```
int QueueLength ( SqQueue Q ){
    return ( (Q.rear – Q.front + MAXQSIZE) % MAXQSIZE );
}
```







3.5.2： 队列的顺序表示和实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

循环队列的操作——循环队列入队

3.5.2： 队列的顺序表示和实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

解决假上溢的方法——引入循环队列

base[0]接在base[MAXQSIZE – 1]之后，若rear+1=M，则令rear=0;

实现方案：利用模（mod ,C 语言中：%）运算。

插入元素：Q.base[Q.rear] = x;

Q.rear = (Q.rear+1)%MAXQSIZE;

3.5.2： 队列的顺序表示和实现

循环队列的操作——循环队列入队

```
Status EnQueue(SqQueue &Q, QElemType e){
    if((Q.rear+1)%MAXQSIZE==Q.front)  return ERROR; //队满
    Q.base[Q.rear]=e;                              //新元素加入队尾
    Q.rear=(Q.rear+1)%MAXQSIZE;                    //队尾指针+1
    return OK;
}
```

3.5.2： 队列的顺序表示和实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

解决假上溢的方法——引入循环队列

base[0]接在base[MAXQSIZE – 1]之后，若rear+1=M，则令rear=0;

实现方案：利用模（mod ,C 语言中：%）运算。

插入元素：Q.base[Q.rear] = x;

Q.rear = (Q.rear+1)%MAXQSIZE;

删除元素：x = Q.base[Q.front];

Q.front = (Q.front+1)%MAXQSIZE;

3.5.2： 队列的顺序表示和实现

循环队列的操作——循环队列出队

```
Status DeQueue(SqQueue &Q, QElemType &e){
        if(Q.front == Q.rear) return ERROR;                    //队空
        e = Q.base[Q.front];                                   //保存队头元素
        Q.front = (Q.front+1)%MAXQSIZE;                        //队头指针+1
        return OK;
}
```

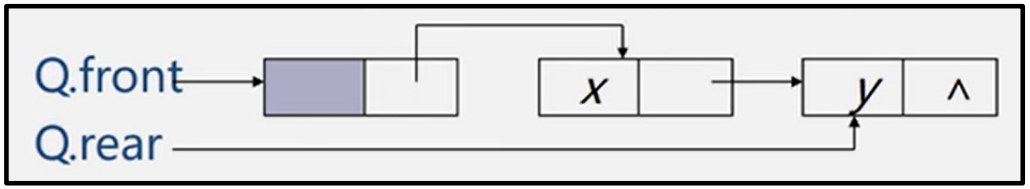
3.5.2： 队列的顺序表示和实现

循环队列的操作——取队头元素

```
SElemType GetHead(SqQuere Q){
    if(Q.front!=Q.rear)      //队列不为空
        return Q.base[Q.front];    //返回队头指针元素
  的值，队头指针不变
```

3.5.3： 链队——队列的链式表达和实现

若用户无法估计队列的长度，则宜采用链队列



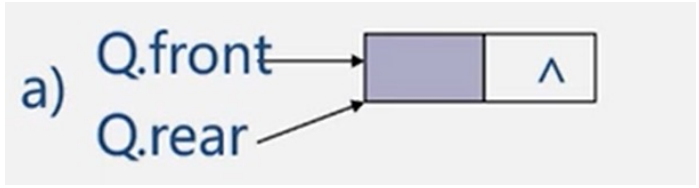
链队列的类型定义

```
#define MAXQSIZE 100 //最大队列长度
typedef struct  Qnode {
      QElemType    data;
      stuct  Qnode  *next;
}QNode, *QuenePtr;
```

```
typedef struct {
    QuenePtr front; // 队头指针
    QuenePtr rear; // 队尾指针
} LinkQueue;
```

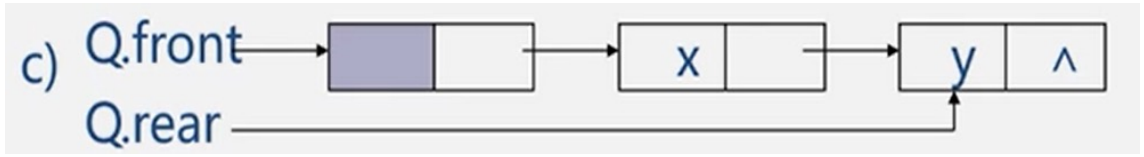
3.5.3： 链队——队列的链式表达和实现

链队列运算指针变化情况

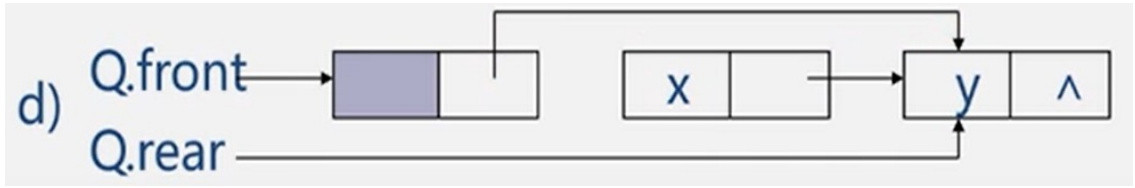


a) 空队列

b) 元素x入队列



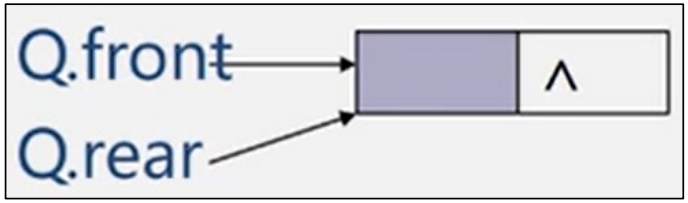
c) y入队列



d) x出队列

3.5.3： 链队——队列的链式表达和实现

链队列的操作——链队列初始化

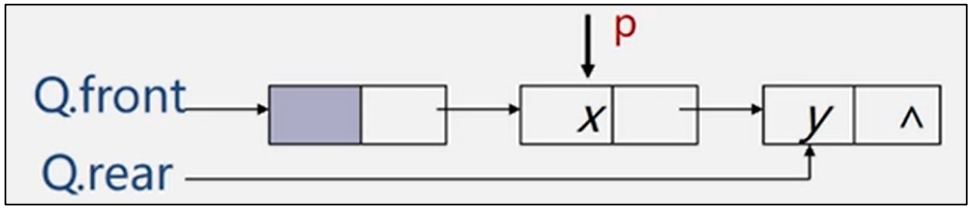



```
Status InitQueue (LinkQueue &Q){
    Q.front=Q.rear=(QueuePtr) malloc(sizeof(QNode));
    if(!Q.front) exit(OVERFLOW);
    Q.front->next=NULL;
    return OK;
}
```

3.5.3： 链队——队列的链式表达和实现

链队列的操作——销毁链队列

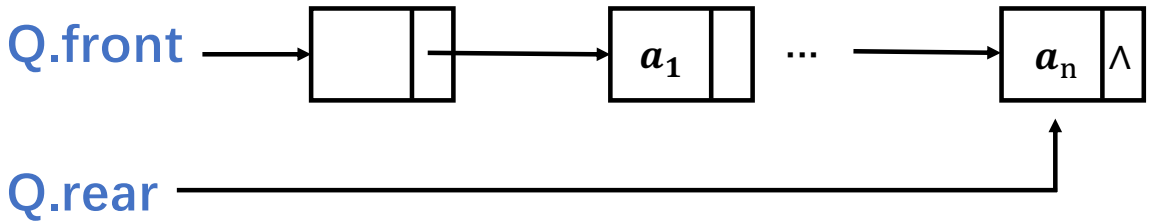
算法思想：从队头节点开始，依次释放所有结点



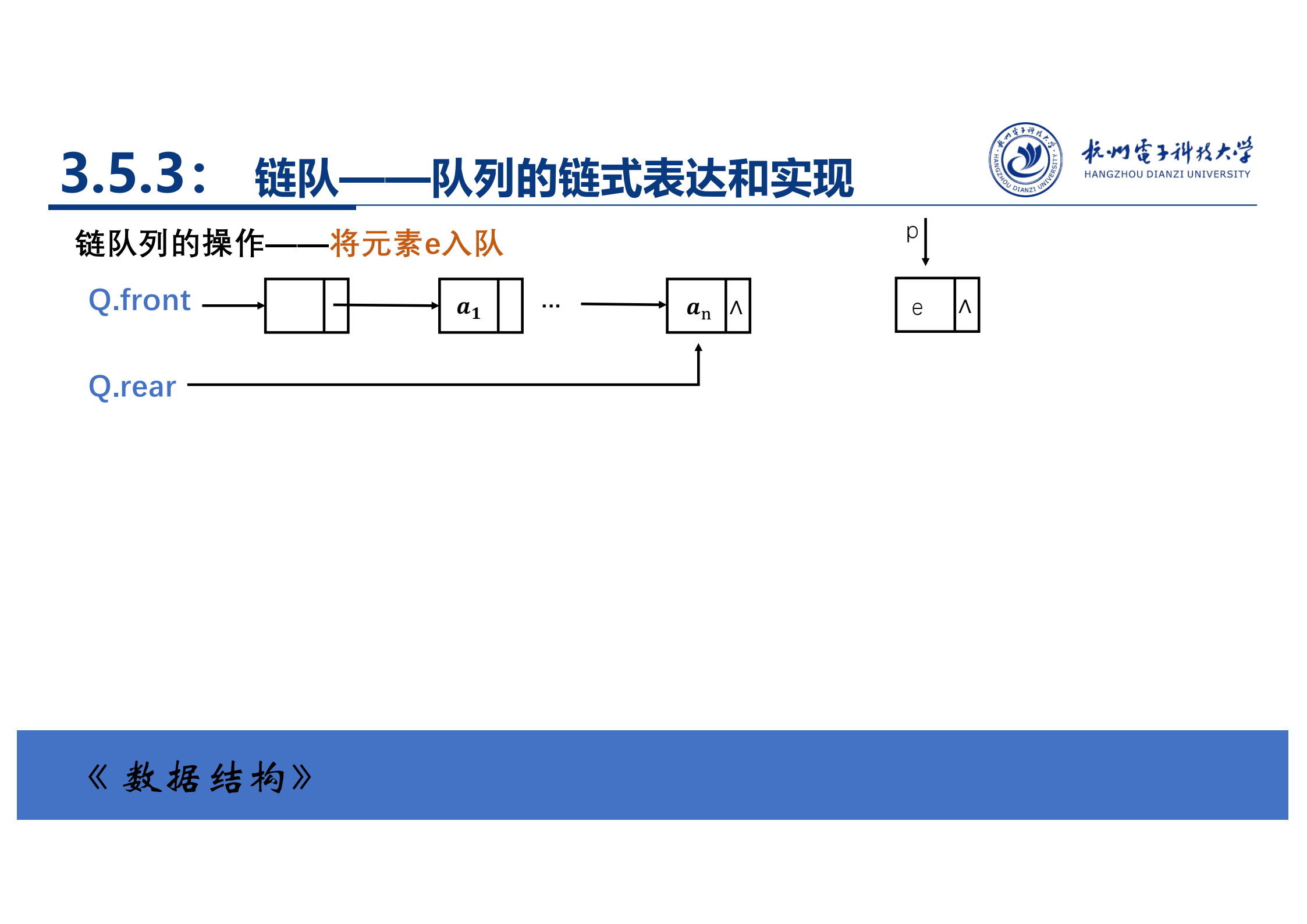

```
Status DestroyQueue (LinkQueue &Q){
  while(Q.front){
        p=Q.front->next;  free(Q.front);  Q.front=p;
    }//Q.rear=Q.front->next;  free(Q.front);  Q.front=Q.rear;
  return OK;
}
```

3.5.3： 链队——队列的链式表达和实现

链队列的操作——将元素e入队



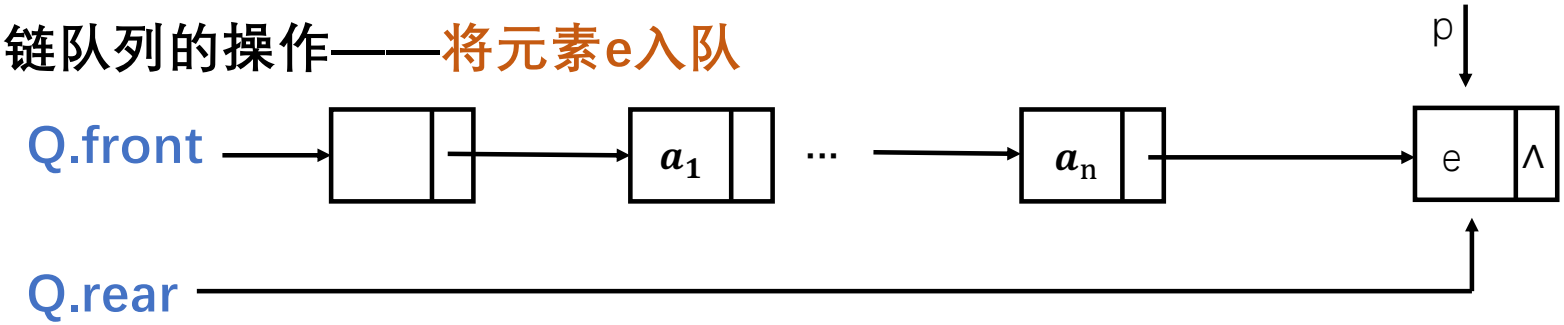
3.5.3： 链队——队列的链式表达和实现



链队列的操作——将元素e入队

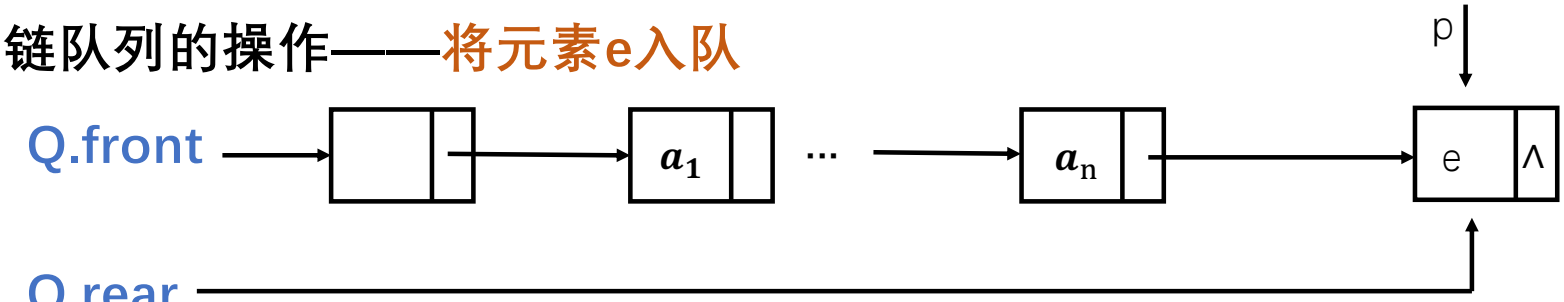
3.5.3： 链队——队列的链式表达和实现

链队列的操作——将元素e入队



3.5.3： 链队——队列的链式表达和实现

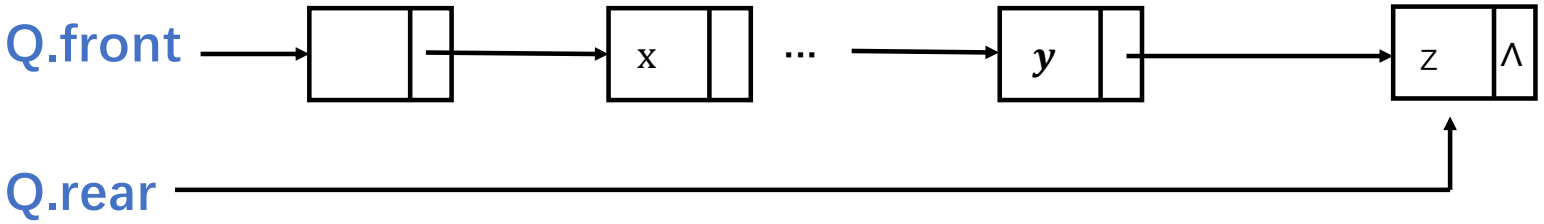
链队列的操作——将元素e入队




```
Status EnQueue(LinkQueue &Q, QElemType e){
    p=(QueuePtr)malloc(sizeof(QNode));
    if(!p) exit(OVERFLOW);
    p->data=e; p->next=NULL;
    Q.rear->next=p;
    Q.rear=p;
    return OK;
}
```

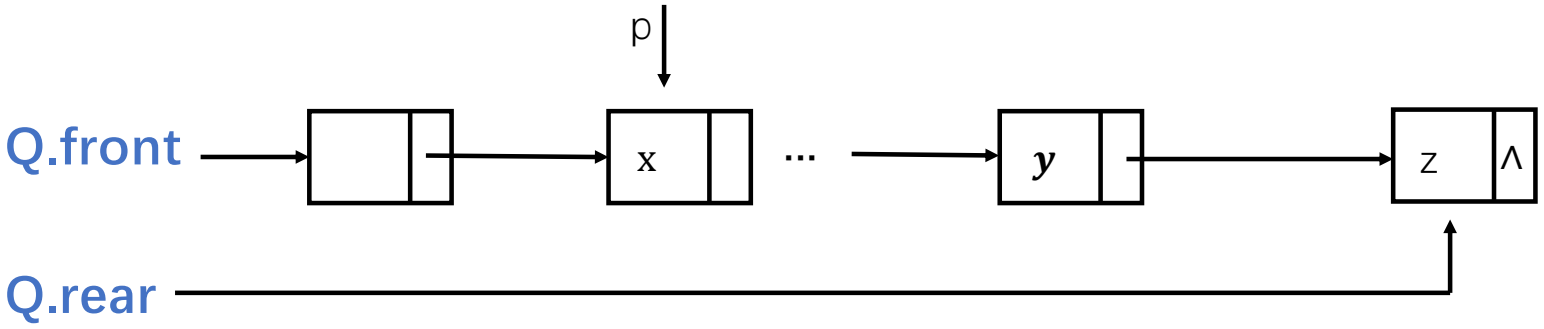
3.5.3： 链队——队列的链式表达和实现

链队列的操作——链队列出队



3.5.3： 链队——队列的链式表达和实现

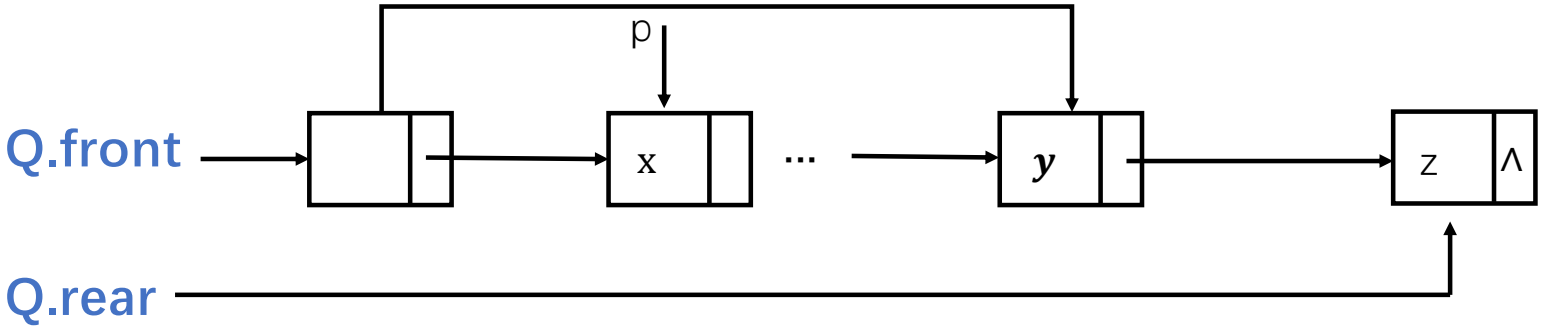
链队列的操作——链队列出队（算法3.18）




```
p = Q.front->next;
```

3.5.3： 链队——队列的链式表达和实现

链队列的操作——链队列出队（算法3.18）



p = Q.front->next;　　e = p->data;

Q.front->next = p->next;

3.5.3： 链队——队列的链式表达和实现

链队列的操作——链队列出队（算法3.18）

P = Q.front->next;　　e = p->data;

Q.front->next = p->next;

free(p);　　/delete p;

3.5.3： 链队——队列的链式表达和实现

链队列的操作——链队列出队（算法3.18）

```
Status DeQueue (LinkQueue &Q,QElemType &e){
  if(Q.front==Q.rear) return ERROR;
  p=Q.front->next;
  e=p->data;
  Q.front->next=p->next;
  if(Q.rear==p) Q.rear=Q.front;
  delete p;
  return OK;
}
```

3.5.3： 链队——队列的链式表达和实现

链队列的操作——求链队列的队头元素（算法3.19）

```
Status GetHead (LinkQueue Q, QElemType &e){
   if(Q.front==Q.rear) return ERROR;
   e=Q.front->next->data;
   return OK;
}
```

随堂测试-判断题

【题1】：若一个栈的输入序列为1，2，3，…，N，输出序列的第一个元素是i，则第j个输出元素是j−i−1。 F

【题2】：序列{1,2,3,4,5}依次入栈，则不可能得到{3,4,1,2,5}的出栈序列。 T

【题3】：栈结构不会出现溢出现象。 F

随堂测试-判断题

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

【题4】：所谓“循环队列”是指用单向循环链表或者循环数组表示的队列。 F

【题5】：可以通过少用一个存储空间的方法解决循环队列假溢出现象。 F

随堂测试-选择题

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

【题1】：设栈S和队列Q的初始状态均为空，元素{1, 2, 3, 4, 5, 6, 7}依次进入栈S。若每个元素出栈后立即进入队列Q，且7个元素出队的顺序是{2, 5, 6, 4, 7, 3, 1}，则栈S的容量至少是：

(A). 1　　(B). 2　　(C). 3　　(D). 4

【题2】：给定一个堆栈的入栈序列为{ 1, 2, …, n }，出栈序列为{ p1, p2, …, pn }。如果p2=n，则存在多少种不同的出栈序列?

(A). 1　　(B). 2　　(C). n-1　　(D). n

随堂测试-选择题

【题3】：若栈采用顺序存储方式存储，现两栈共享空间V[m]：top[i]代表第i（i=1或2）个栈的栈顶；栈1的底在V[0]，栈2的底在V[m-1]，则栈满的条件是：

(A). |top[2]-top[1]|==0　　**(B). top[1]+top[2]==m**

(C). top[1]==top[2]　　**(D). top[1]+1==top[2]**

【题4】：如果循环队列用大小为m的数组表示，队头位置为front、队列元素个数为size，那么队尾元素位置rear为：

(A). front+size　　**(B). front+size-1**

(C). (front+size)%m　　**(D). (front+size-1)%m**

随堂测试-选择题

【题5】：循环队列的队满条件为 ( )。

(A). (sq.rear+1) % maxsize ==(sq.front+1) % maxsize

(B). (sq.front+1) % maxsize ==sq.rear

(C). (sq.rear+1) % maxsize ==sq.front

(D). sq.rear ==sq.front

【题6】：循环队列的引入，目的是为了克服( )。

(A). 假溢出问题　(B). 真溢出问题　(C). 空间不够用　(D). 操作不方便

随堂测试-选择题

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

【题7】：已知初始为空的队列 Q 的一端仅能进行入队操作，另外一端既能进行入队操作又能进行出队操作。若 Q 的入队序列是 1、2、3、4、5，则不能得到的出队序列是：

(A). 5、4、3、1、2

(B). 5、3、1、2、4

(C). 4、2、1、3、5

(D). 4、1、3、2、5